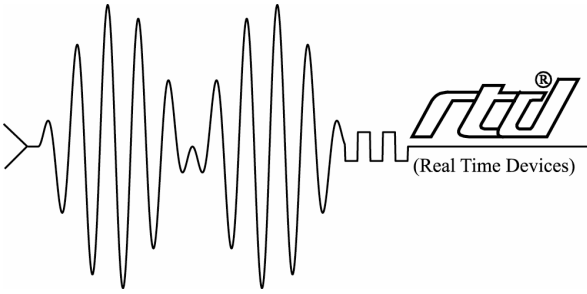
DM6425 Driver for Windows 2000/XP

Version 1.0

User's Manual



RTD Embedded Technologies, Inc.
"Accessing the Analog World"®

SWM-640010026
Rev. A

ISO9001 and AS9100 Certified

RTD Embedded Technologies, INC.
103 Innovation Blvd.
State College, PA 16803-0906

Phone: +1-814-234-8087
FAX: +1-814-234-5218

E-mail
sales@rtd.com
techsupport@rtd.com

web site
http://www.rtd.com

Revision History

03/22/2006	Rev A (Version 1.0) released

DM6425 Driver for Windows 2000/XP
Published by:

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College, PA 16803-0906

Copyright 2006 RTD Embedded Technologies, Inc.
All rights reserved
Printed in U.S.A.

The RTD Logo is a registered trademark of RTD Embedded Technologies. cpuModule and utilityModule are trademarks of RTD Embedded Technologies. PS/2, PC/XT, PC/AT and IBM are trademarks of International Business Machines Inc. MS-DOS, Windows, Windows 98, Windows NT, Windows 2000 and Windows XP are trademarks of Microsoft Corp. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

Table of Contents

TABLE OF CONTENTS **4**

INTRODUCTION **5**

INSTALLATION **6**

- INSTALLATION OF THE DRIVER AND EXAMPLE PROGRAMS 6
- THE DM6425 BOARD DRIVER 7
- INSTALLING ADDITIONAL DM6425 BOARDS IN THE SYSTEM 7
- USING WINDOWS REGISTRY KEYS 8

THE DRIVER API FUNCTIONS **9**

DM6425 FEATURES **10**

- MEASUREMENT SCENARIOS 10
- CHANNEL-GAIN CIRCUITRY 11
- INTERRUPTS 14
- DIRECT MEMORY ACCESS (DMA) 15

EXAMPLE PROGRAMS REFERENCE **16**

- WIN32 CONSOLE APPLICATIONS 16
- WIN32 WINDOWS APPLICATIONS 16

LIMITED WARRANTY **18**

Introduction

The DM6425 data acquisition board Windows 2000/XP driver is designed for programmers who write Windows-based application programs using RTD's DM6425 board.

The driver provides an Application Programming Interface with a variety of function calls for performing all of the board's data acquisition tasks.

The board driver is based on BlueWater System's WinRT device driver kit.

There are example programs to demonstrate the various board features and the usage of the driver API. The example programs are compiled using Microsoft Visual C++ ver. 6.0.

Installation

Installation of the Driver and Example Programs

Before installing the driver and example program files, you need to install the DM6425 board in your PC. Please follow the instructions of the manufacturer, of how to install the board in your computer.

To install the drivers for the DM6425 you need the Windows Software (DM6425). The readme.txt file in this package describes the necessary installation steps.

Also in the software package is the **Setup.exe** program, which installs on your PC the DM6425 board driver and example programs. The setup program automatically detects your operating system and installs the appropriate files on your PC.
After starting the setup, please follow the instructions on the screen to install the programs. You can select the directory where to install the files.
The setup also adds to the 'Start menu' under the 'Programs' folder of your Windows system the 'RTD Embedded Technologies' folder. It contains shortcuts to the example programs, the driver configuration utility and the readme.txt file.

The example programs will be installed in the target directory under the 'Examples' folder. This folder contains the example program sources for the DM6425 board, and the project files for Microsoft Visual C++ users to rebuild the programs. Pre-built versions of the examples are in the 'Examples\!Exe' folder. The example programs are compiled with the Microsoft Visual C++ ver. 6.0. In case of different version of Visual C is installed on your PC, please rebuild the executable files.

Uninstallation: you can uninstall the DM6425 driver and example programs under the 'Control panel' with the 'Add/Remove Programs' tool.

The DM6425 board driver

RTD's DM6425 board driver handles the hardware through BlueWater's WinRT driver. The WinRT driver provides the low-level access to the board, and the RTD's Drvr6425.dll provides the device API for the programmers, and communicates with the hardware through the WinRT driver.

The RTD board drivers allow you to use multiple boards in the system at the same time.

The base address of the DM6425 board can be set by jumpers. If you are using multiple boards in your system, you need to assign a different base address to each board. The IRQ and DMA channels are set programmatically. When you install a WinRT driver for the board, you must select the IRQ and DMA channels, and use these during the operation of the board.

Each board requires a unique WinRT device to handle the hardware. Also the second DMA channel of the DM6425 board requires a unique device.

The DM6425 board has a dual-DMA feature (using two DMA channels in parallel). This operating mode requires a different driver device for the second DMA channel. So every DM6425 board uses two WinRT devices one after the other, and they are using successive driver ID's.

Here is an example of a set of resource allocations for a DM6425:
Board 1: 0x300 base address (0x700 for the second IO range and 0xB00 for the third), DMA channel 5, DMA buffer size 0x4000 and IRQ 11
Board 1's 2nd DMA channel: DMA channel 6, DMA buffer size 0x4000.

These settings can be configured in the Control Panel/System/[Hardware]/Device Manager.

Installing additional DM6425 boards in the system

It is possible to use more than one board in the system. To install additional boards simply follow the steps described in readme.txt, located in the Windows Software (DM6425) package.

Using Windows registry keys

The RTD drivers are using the following registry keys:

Windows 2000/XP device class information:

```
HKEY_LOCAL_MACHINE
|
| -- System
     |
     | -- CurrentControlSet
          |
          | -- Control
               |
               | -- Class
                    |
                    | -- {D695ED6A-630D-4D83-D8B-F1F0AC107AD0}
                    |
                    | -- 0001
```

Windows 2000/XP Plug and Play devices are enumerated under the Enum key:

```
HKEY_LOCAL_MACHINE
|
| -- System
   |
   | -- CurrentControlSet
        |
        | -- Enum
             |
             | -- PCI
```

The Driver Api Functions

The resources on the DM6425 board can be accessed from Windows through the driver API (Application Programming Interface) functions. The executable code of these functions is located in the Drvr6425.dll file. To write applications using the API functions you must include the Reg6425.H header file, and link the program with the Drvr6425.lib import library file.
In the example programs you can find different examples of how to use the driver. Also, each API function is fully documented in the Reg6425.H file.

DM6425 Features

Measurement Scenarios

Through selecting different options for A/D Conversion Trigger (SetConversionSelect6425), Burst Clock Start Trigger (SetBurstTrigger6425), and creating different Channel-Gain Tables, you have innumerable sampling scenarios. The following bullets try to enumerate only the most frequently used measurement setups.

- ***Single Conversion***

  In this mode, a single channel is sampled whenever StartConversion6425 is called. The Channel Gain Latch (see SetChannelGain6425) specifies the channel to sample. This is the easiest scenario of all. It can be used in a variety of applications, such as sample every time a key is pressed on the keyboard, sample with each iteration of a loop, or watch the system clock and sample every five seconds.

- ***Multiple Conversions***

  In this mode, conversions are continuously performed at the rate of the Pacer Clock, or other selected A/D Conversion Signal rate. The pacer clock can be internal or external. The maximum rate supported by the board is 500KHz. If you use the internal pacer clock, you must program it to run at the desired rate (SetPacerClock6425).

  This mode is ideal for filling arrays, acquiring data for a specified period, and taking a specified number of samples.

- ***Random Channel Scan***

  In this mode, the Channel-Gain Table is incrementally scanned through, with each selected A/D Conversion Signal pulse starting a conversion at the channel and gain specified in the current table entry. Before starting a conversion sequence Channel Gain Table, you need to load the table with the desired data. Then make sure that the Channel-Gain Table is enabled by the function EnableTables6425. This enables the A/D and Digital portion of the Channel Gain Table as well. Each rising edge of selected A/D Conversion Signal starts a conversion using the current Channel Gain data and then increments to the next position in the table. When the last entry is reached, the next pulse starts the table over again.

- ***Programmable Burst***

  In this mode, a single trigger initiates a scan of the entire Channel-Gain Table. Before starting a burst of the Channel-Gain Table, you need to load the table with the desired data. Then enable the Channel-Gain Table by EnableTable6425.

  Burst is used when you want one sample from a specified number of channels for each trigger. The burst trigger starts the Burst Clock and the Burst Clock initiates each conversion. At high speeds, the burst mode emulates simultaneous sampling of multiple input channels. For time critical simultaneous sampling applications, a simultaneous sample-and-hold board can be used (SS8 eight-channel boards are also available from RTD Embedded Technologies).

- ***Programmable Multi-Scan***

  This mode - when the A/D Conversion Start Signal is the Burst Clock - lets you scan the Channel Gain Table after a Burst Clock Start Signal. When the Channel Gain Table is empty, the Burst Clock is stopped, and will wait for a new Start Signal.

Channel-Gain Circuitry

Channel-Gain Tables are traditionally for implementing random channel scan analog input on boards where a single A/D converter is multiplexed for 8, 16 or more analog input channels.

The Channel-Gain Circuitry embeds a 1024x24 bit on-board memory (Channel Scan Memory), called a *Channel-Gain Table* (CGT) for historical reasons. Every 24-bit row (entry) in a CGT is an instruction executed by the Channel-Gain circuitry. Execution happens at a programmable rate. Channel-Gain Latch (CGL), provided for easy, single channel analog input, can be perceived as a special, single row CGT for the following description. Unless explicitly indicated, explanation holds for the CGL, as well.

The table below pictures the format of a CGT entry:

DO	Skip	Pause	Se/Diff	Range	Gain	Channel
8 bits	1 bit	1 bit	1 bit	2 bits	3 bits	5 bits

Channel **Analog Input Channel**

Specifies the Analog Input channel to sample. Depending on your configuration you may have 16 differential channels (AIN1…AIN16), or 32 Single-Ended channels (AIN1…AIN32)

Gain **Analog Input Gain**

This field specifies the gain to apply to the input. Available choices are1x, 2x, 4x, 8x.

Range **Analog Input Range**

Specifies one of the three supported input ranges: ±5 Volts, ±10 Volts or 0…10Volts.

Se/Diff **Analog Input Type**

Pause **Pause Bit**

If this bit is enabled by the SetADPauseEnable6425 function, execution of the Channel-Gain Table stops *after* executing this entry. Execution is resumed with the next CGT entry when the programmed Pacer Clock start trigger occurs.

EXAMPLE: Pause Bit can be used when you have two sequences of entries, each to be executed on a different event (trigger). Suppose that CGT is driven by the Pacer Clock, and the Pacer Clock is started on the External Trigger. The External Trigger comes from a device, whose pulses indicate two different events. Odd pulses indicate an event, on which you want to react by sampling AIN1 and AIN2, on even pulses you want to sample AIN3, AIN4 and AIN5. In this case, you would create a 5 entry CGT:

Entry #1: AIN1, Pause Bit = 0
Entry #2: AIN2, Pause Bit = 1
Entry #3: AIN3, Pause Bit = 0
Entry #4: AIN4, Pause Bit = 0
Entry #5: AIN5, Pause Bit = 1

In this case, the first pulse on the External Trigger line starts executing the CGT at the rate of the Pacer Clock. After executing the first two entries, execution stops and is waiting for the next External Trigger pulse. The second pulse resumes execution, and entries #3, #4 and #5 are executed at the rate of the Pacer Clock. Execution pauses again, after executing entry #5. A third External trigger pulse continues execution with entry #1, and so on.

NOTE: When the Channel-Gain Latch is used, or in burst mode, Pause Bit is ignored.

Skip **Skip Bit**

When the Skip Bit is set, the entry is skipped, which means that the A/D conversion is performed but the resulting sample is not written into the A/D FIFO. This feature provides a way to sample multiple channels at different rates without saving unwanted data.

EXAMPLE: In this example, we want to sample AIN1 in every second and AIN4 in every three seconds. For this end, we must create CGT with six entries:

Entry #1: AIN1, Skip Bit = 0
Entry #2: AIN4, Skip Bit = 1
Entry #3: AIN1, Skip Bit = 0
Entry #4: AIN4, Skip Bit = 1
Entry #5: AIN1, Skip Bit = 0
Entry #6: AIN4, Skip Bit = 0

Next, we set the Pacer Clock to run at 2 Hz (0.5 seconds). This allows us to sample each channel once per second, the maximum sampling rate required by one of the channels (pacer clock rate = number of different channels sampled x fastest sample rate).

The first Pacer Clock pulse starts an A/D conversion according to the parameters set in the first entry of the Channel-Gain Table, and each successive clock pulse incrementally steps through the table entries. The first clock pulse takes a sample on AIN1. The second pulse looks at the second entry in the table and sees that the Skip Bit is set. Sample is taken, but is not stored in the FIFO. The third pulse takes a sample on AIN1 again, the fourth pulse skips the next entry, and the fifth pulse takes our third reading on AIN1. On the sixth pulse, the Skip Bit is disabled, AIN4 is sampled and sample is stored to the FIFO. Then the sequence starts over again with entry #1. Samples are not stored when they are not wanted, saving memory and eliminating the need to throw away unwanted data.

NOTE: When the Channel-Gain Latch is used, Skip Bit is ignored.

DO **8-Bit Digital Table**

The digital portion of the Channel-Gain Table, also referred to as Digital Table, can be used to control input expansion boards such as the TMX32 Analog Input Expansion board. The expansion board is driven at the same speed as the A/D conversions are performed, with no software overhead.

EXAMPLE: Let us consider the following simple example on driving an analog input expansion board.

In this example, we have a TMX32 expansion board connected to AIN1 on the DM6425. We have three signals to sample, one is connected to the first channel of the expansion board (EAIN1), the second is connected to the fourth channel of the expansion board (EAIN4) and the third is connected directly to AIN2 of the DM6425.

We need to create the following Channel-Gain Table:

Entry #1: AIN1, gain=1, DO=0
Entry #2: AIN1, gain=4, DO=3
Entry #3: AIN2, gain=1, DO=3

Execution, starting with entry #1, samples AIN1 and simultaneously outputs 0 on Digital Port 1. This will cause the expansion board to switch to EAIN1.

Entry #2 will sample AIN1, which is now connected to EAIN1, and simultaneously outputs 3 on Digital Port 1. As a result, the expansion board switches to EAIN4.

Next, entry #3 samples AIN2 and outputs 3 on Digital Port 1, which makes the expansion board to switch (again) to EAIN4.

When executing entry #1 again, AIN1 is sampled which is now connected to EAIN4, and so on.

NOTE: If you only need to use the A/D part of the table, you do not have to program the Digital Table. However, if you only want to use the Digital part of the table, you must program the A/D part of the table.

NOTE: When the Channel-Gain Latch is used, Digital Table is ignored.

When using the Channel Gain Table, you should group your entries to maximize the throughput of your module. Low-level input signals and varying gains are likely to drop the throughput rate because low level inputs must drive out high level input residual signals. To maximize throughput:

- Keep channels configured for a certain range grouped together, even if they are out of sequence.
- Use external signal conditioning if you are performing high speed scanning of low level signals. This increases throughput and reduces noise.
- If you have room in the channel-gain table, you can make an entry twice to make sure that sufficient settling time has been allowed and an accurate reading has been taken. Set the skip bit for the first entry so that it is ignored.
- For best results, do not use the channel-gain table when measuring steady-state signals. Use the single convert mode to step through the channels.

Interrupts

The controller can receive interrupt requests from up to 20 sources. These 20 sources cover the most important internal signals of the board plus 2 external signals:

- A/D Sample Counter Countdown

  Interrupt is generated when the A/D Sample Counter counts down to zero.
  This interrupt can be used to count more than 65535 samples by counting the turnovers of the Sample Counter.

- A/D Start Convert

  Interrupt is generated when a conversion is started.

- A/D End Of Convert

  Interrupt is generated when an end of convert is issued by the A/D converter.

- A/D FIFO Write

  Interrupt is generated when sample enters the A/D FIFO.
  This interrupt can be used for reading and processing samples real-time.

- A/D FIFO Half Full

  Interrupt is generated when the A/D FIFO is half full.

- A/D DMA Done

  Interrupt is generated when the A/D DMA done flag goes high.

- CGT Reset

  Interrupt is generated when the Channel-Gain Table recycles execution to the first table entry.
  This interrupt can be used for reading and processing a burst of samples from different channels in real-time.

- CGT Pause

  Interrupt is generated when Channel-Gain Table execution is paused waiting for a new trigger.

- External Pacer Clock

  Interrupt is generated when the external pacer clock line is pulsed.

- External Trigger

  Interrupt is generated when the external trigger line is pulsed.

- Digital Interrupt, Port 0

  Interrupt is generated when the Advanced Digital Trigger signals a Digital Interrupt.
  This interrupt can be used to detect (and react to) patterns on Digital Input Port 0.

- User Timer/Counter 0 Out

- User Timer/Counter 0 Out, inverted

- User Timer/Counter 1 Out

  Interrupt is generated on the ticks of User T/C0 (i.e., when the counter counts down to zero).
  This interrupt gives you a general-purpose means of measuring real time, frequency, or counting events. It is also intended to use for Pulse output generation.

- Digital Input FIFO Half Full

  Interrupt is generated when the Digital Input FIFO is half full.

- Digital Input FIFO Write

  Interrupt is generated when sample enters the Digital Input FIFO.

- Digital Interrupt, Port 2

  Interrupt is generated when the Advanced Digital Trigger signals a Digital Interrupt. This interrupt can be used to detect (and react to) patterns on Digital Input Port 2.
- Analog Threshold

  Interrupt is generated when Analog Threshold condition is met.
- External Interrupt 1

  Interrupt is generated when the External Interrupt 1 line is pulsed.
- External Interrupt 2

  Interrupt is generated when the External Interrupt 2 line is pulsed.

It is easy to service interrupts with the DM6425 driver.

Actually, when you install your interrupt handler, it is not a real Interrupt Service Routine, since you can not do that under Windows. Your handler is an ordinary routine, which is called by the real Interrupt Service Routine implemented by the DM6425 driver. The driver's Interrupt Service Routine schedules your interrupt handler for execution as a separate thread, acknowledges the interrupt to the board's interrupt controller and returns. After returning, the scheduled thread starts executing your interrupt handler.

Direct Memory Access (DMA)

DMA transfers data between peripheral device and PC memory without using the processor as an intermediate. This method allows very fast data transfer rates.

The DM6425 driver provides an interface for the user to handle the DMA channels of the board. The driver handles the dual-DMA mode of the DM6425, too.

The following program steps are necessary to use the DMA:

- Initialize the board, setup triggering modes.
- Install the DMA handler.

  InstallDMA6425(DEVICE_NO, &dmasetup, TRUE, 0, 0);
- Program DMA channel.

  SetADDMA6425(DEVICE_NO,ADDMAChannel,0);
- Perform data collection.
- Close DMA channel.

  DeInstallDMA6425(DEVICE_NO, 0);

To see how to use the DMA channel on the DM6425 board, see the dma and wdma example programs.

Example Programs Reference

Win32 console applications

Name	Feature	Remarks
dma	Single DMA channel acquisition Sampling on pacer clock Start trigger: software Stop trigger: software	The data is displayed numerically on the screen.
speedtst	Sampling on pacer clock Start trigger: software Stop trigger: software Using IRQ handler function	It demonstrates three methods of data acquisition with interrupt: • callback function • poll the IRQ counter • get data with autoincrement mode

Win32 Windows applications

Name	Feature	Remarks
highspeed	High speed data acquisition program.	It demonstrates a multi-channel DAQ with storing the samples in a large memory buffer and save it on disk when the buffer is full.
wanlgtrgr	Analog Thresholds	The user can change the threshold voltage with the on-screen slider.
wdac	Digital/analog conversation.	The user can change the output voltage with the on-screen slider.
wdigital	It programs the digital port 0 to input, port 1 to output.	The user can set the digital output port; the program reads the input port.
wdma	Single DMA channel acquisition. Sampling on pacer clock Start trigger: software Stop trigger: software	The data displayed graphically on the screen.
wdualdma	Dual DMA channel acquisition. Sampling on pacer clock Start trigger: software Stop trigger: software	It demonstrates the high speed gap-free data acquisition. The data stored in file.
whidin	High-speed digital input example. The user timer-counter is programmed to sample the digital input. At digital FIFO half full generated an interrupt, and the data is stored and displayed.	The data displayed graphically and stored in file.
wintrpts	The user timer-counter is programmed to generate	The data displayed graphically.

	interrupts. The analog input sampled when the IT occurs.	
wrandom	Random channel acquisition with CGT. The pacer clock programmed to start conversion with burst clock.	The samples are displayed numerically.
wrepinsw	Sampling on pacer clock. Start trigger: software Stop trigger: software Interrupt on sample counter	On sample counter IT the program reads and displays the samples graphically.
wsmpcnt	Sampling on pacer clock. Start trigger: software Stop trigger: sample counter. Interrupt on sample counter	On sample counter IT the program reads and displays the samples graphically. The user can repeat the measure by pressing a key.
wsofttrig	Single A/D sampling. Start trigger: software Stop trigger: software	The user can sample an analog input by pressing a button. The sample displayed on the screen in Volts.
w2board	Example on how to use two boards.	Demonstration of the driver's multi-board feature.
wtimers	Demonstration of user timer-counters.	The counters are programmed to count the elapsed time in seconds.

Limited Warranty

RTD Embedded Technologies, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from RTD Embedded Technologies, INC. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, RTD Embedded Technologies will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to RTD Embedded Technologies. All replaced parts and products become the property of RTD Embedded Technologies. Before returning any product for repair, customers are required to contact the factory for an RMA number.

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, "acts of God" or other contingencies beyond the control of RTD Embedded Technologies), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN RTD Embedded Technologies. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND RTD Embedded Technologies EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER'S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL RTD Embedded Technologies BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College PA 16803-0906
USA
Our website: www.rtd.com